Methodol Comput Appl Probab
DOI 10.1007/s11009-016-9534-3

Queues with Dropping Functions and Autocorrelated Arrivals

Pawel Mrozowski[1] · **Andrzej Chydzinski**[2]

Received: 14 January 2016 / Revised: 25 September 2016 / Accepted: 4 December 2016

Abstract We present an analysis of the queueing system in which arriving jobs are dropped with probability depending on the queue size. The arrivals are assumed to be autocorrelated and they are modeled by the Markov-modulated Poisson process. Both transient and stationary distributions of the queue size, as well as the system loss ratio and throughput are obtained. The analytical results are accompanied with numerical examples based on the autocorrelated traffic recorded in an IP computer network.

Keywords Queueing system · Dropping function · Markov-modulated Poisson process · Queue size distribution · Active queue management

Mathematics Subject Classification (2010) 60K25 · 68M20 · 90B22 · 90B18

1 Introduction

The queue with the dropping function is a simple FIFO queue with an additional mechanism. Namely, an arriving job (customer, packet etc.) can be dropped (rejected) with probability $d(n)$, where n is the queue size observed upon arrival of this job (see Fig. 1). The function $d(n)$ is called a dropping function.

The significance of the queueing system with the dropping function can be presented from two perspectives:

(a) universal sense of the system and its general applicability,
(b) direct applicability of the system in networking.

✉ Andrzej Chydzinski
Andrzej.Chydzinski@polsl.pl

[1] Avio Polska Sp. z o.o. Grazynskiego 141, Bielsko-Biala 43-300, Poland

[2] Institute of Informatics, Silesian University of Technology, Akademicka 16, 44-100 Gliwice, Poland

Published online: 20 December 2016

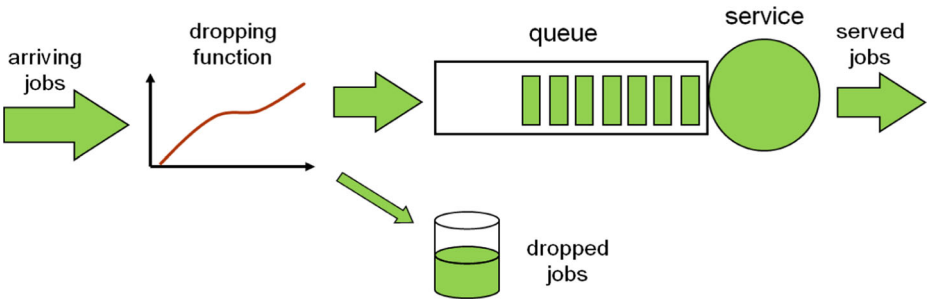

Fig. 1 Queueing system with the dropping function

As regards (a), it is well known that in the classic FIFO queueing model we cannot control the performance. Given the arrival and service processes, we can calculate characteristics like the queue length, system throughput, number of losses, etc., but we cannot control them. On the other hand, in some applications of queueing systems there is a need to control the performance of the queue, namely to set the mean queue size or the throughput etc. to an arbitrary value.

There are at least three ways to achieve that. Firstly, we may try to alter dynamically arrival rate, depending on the current or past system state. Secondly, we may try to alter dynamically the service rate. Thirdly, we may try to reject arriving jobs. The latter is similar to alternating the arrival rate, but the significant difference is that dropping jobs causes losses, i.e. jobs that were not served and never return to the queue.

There are several queueing models of the first and the second type analyzed in the literature. For example, threshold-based queueing models are studied in (Chydzinski 2002; Pacheco and Ribeiro 2008; Bekker 2009). In such models, the arrival or service rates alternate when the queue length reaches a threshold level.

Intuitively, the third approach is the simplest one – it is usually a simple matter to drop an arriving job. Furthermore, application of the dropping function enables a powerful control on the performance of the queueing system. For instance, in Chydzinski and Chrost (2011) it was shown that using dropping functions allows setting the average queue size. The same applies to other parameters, e.g. the queue size variance, the system throughput, etc.

Queueing systems with dropping functions have at least one direct application, namely the active queue management in Internet routers. It has been shown that simple FIFO tail-drop queues, commonly used in Internet routers by device vendors, have some important disadvantages. In particular, they cause large queueing delays, flow synchronization and unfairness between flows. To overcome these problems, the active queue management (AQM) for Internet routers was proposed. The idea was that the router can drop incoming packets even if the buffer is not full yet, thus preventing the queue from building up.

The router can drop incoming packets depending on several factors, but the simplest approach is that the incoming packet is dropped with the probability that is a function of the queue size.

Now, it must be stressed that the analytical results obtained so far for queues with dropping functions fed by classic queueing traffic models (e.g. Poisson) are of little use when the real arrival process is autocorrelated. This is the case of networking – it is well known that Internet traffic possesses strongly autocorrelated structure. In Fig. 2 an example of the

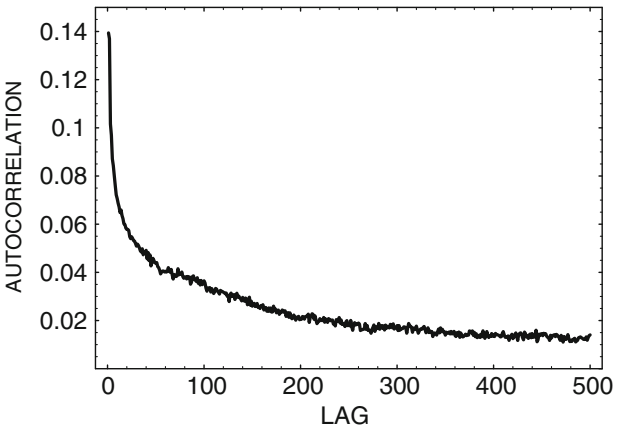

Fig. 2 Example of autocorrelation in IP traffic

interarrival times autocorrelation function based on recorded traffic is presented[1]. As we can see in the figure, the autocorrelation decays slowly and is significant on several time scales.

Not taking this autocorrelation into account may lead to extreme underestimation (in a negative way) of characteristics of the queueing system. An example can be found in Fig. 2 of Chydzinski (2006a), where a comparison of computed full buffer probabilities in FIFO queues is presented. Both queues have the same buffers, arrival rates and service rates, but one of them uses uncorrelated arrivals (Poisson), whilst the other uses autocorrelated arrivals with the autocorrelation function as in Fig. 2. The obtained full buffer probability for the Poisson traffic is 8.97×10^{-33}, while the value obtained for the autocorrelated traffic is 6.55×10^{-3}. Therefore, not taking into account the autocorrelation resulted in the optimistic underestimation by 30 orders of magnitude.

For these reasons, in this paper we carry out an analysis of the queueing system with the dropping function and autocorrelated arrivals. We do not impose any assumptions on the dropping function nor the service time distribution – they both can have any form.

In order to make the results useful in practice, the following two requirements on the arrival process must be met: it has to be able to mimic arbitrary shape of the autocorrelation function and there should exist an algorithm, which allows fitting the model parameters to this particular shape.

Both these requirements are fulfilled by the Markov-modulated Poisson process (MMPP), Fischer and Meier-Hellstern (1992). In particular, the MMPP is analytically tractable, allows for precise fitting of complicated shapes of the autocorrelation function (see e.g. Salvador et al. (2003)) and several MMPP parameter fitting procedures have been proposed to date, (Salvador et al. 2003; Deng and Mark 1993; Ryden 1996; Yoshihara et al. 2001; Singh and Dattatreya 2004). Therefore, it will be used as the arrival traffic model.

The paper is organized as follows. In Section 2, references to papers on queues with dropping functions, the MMPP process, the MMPP queue and the active queue management are given. In Section 3, the definition and basic formulas on the MMPP arrival process

[1]The traffic was recorded within the Passive Measurement and Analysis Project, trace file FRG-1137208198-1.tsh.

are recalled. In Section 4, the queueing model is formally defined. Then, in Section 5, the results on transient and stationary queue size distributions, as well as the loss ratio, are presented. Section 6 is devoted to calculation of the counting function of the arrival process filtered by the dropping function – this is needed to use the results of Section 5 in practice. In Section 7, some numerical examples are presented. In particular, the examples demonstrating the abilities of the dropping function to maintain a given average queue size and a given throughput are shown. The final conclusions are gathered in Section 8.

2 Related Work

Queueing systems with dropping functions and uncorrelated arrivals were studied in the following papers. In Bonald et al. (2000), an approximate analysis of the model with batch Poisson arrivals, linear dropping function and exponential service time distribution was presented. In Hao and Wei (2005), an approximate analysis of the model with general batch arrivals and exponential service times was carried out. In Chydzinski and Chrost (2011) an exact, steady-state analysis of the model with arbitrary dropping function, Poisson arrivals and arbitrary service time was performed. In Kempa (2013a) an exact analysis of the model with arbitrary dropping function, general uncorrelated arrivals and exponential service time was presented. In Kempa (2013b), the transient analysis of the model with arbitrary dropping function and Poisson arrivals was carried out. Finally, in Tikhonenko and Kempa (2013) the system with the Poisson arrival stream and a general distribution of the job size has been solved.

As for the arrival process, we refer the reader to the excellent MMPP cookbook, Fischer and Meier-Hellstern (1992), and the references given there. In this cookbook, the main results on the infinite-buffer MMPP queue (without the dropping function), are presented as well. The finite-buffer MMPP queue in the steady state was analyzed in Baiocchi and Blefari-Melazzi (1992). The same system in the transient state was studied in Chydzinski (2006a).

Regarding the active queue management in routers, the famous RED algorithm, Floyd and Jacobson (1993), was the first exploiting the dropping function. In the case of RED, the dropping function is the simplest possible, i.e. the linear function. Besides the linear function, some other dropping functions were used in literature: the exponential dropping function, Athuraliya et al. (2001), or the doubly linear dropping function, Rosolen et al. (1999). The active queue management is a widely studied subject. In addition to such well-recognized algorithms as REM (Athuraliya et al. 2001), PI (Hollot et al. 2002; Wu et al. 2001), BLUE (Feng et al. 2002), GREEN (Wydrowski and Zukerman 2002) and AVQ (Kunniyur and Srikant 2004), several other propositions emerged recently, e.g. (Na et al. 2012; Suzer et al. 2012; Farzaneh et al. 2013; Kahe et al. 2013). The new algorithms are usually evaluated either by means of simulators (ns2, ns3), or by means of the control theory. We prefer a different approach, based on tools and results of queueing theory.

The methodology used herein is an extension of the methodology used in solving classic queues with Markovian arrival processes (see e.g. Chydzinski 2006a, 2006b, 2006c), based on formulating and solving a set of Volterra integral equations in the convolution form. The main difference and difficulty herein is a complicated characterization of the counting function of the arrival process filtered by the dropping function, which requires a different approach.

3 Arrival Process

The Markov-modulated Poisson process, Fischer and Meier-Hellstern (1992), is constructed by varying the rate of Poisson arrivals according to a continuous-time Markov chain, called the modulating chain. Assuming that the state space of the modulating chain is $\{1, \ldots, m\}$, to parameterize an MMPP we have to provide the intensity matrix of the modulating chain, $\mathbf{Q}$, and the vector of corresponding intensities of Poisson arrivals, $[\lambda_1, \ldots, \lambda_m]$. The latter will also be used in the form of a diagonal $m \times m$ matrix:

$$\mathbf{\Lambda} = diag[\lambda_1, \ldots, \lambda_m].$$

The average rate of the MMPP can be obtained as

$$\lambda = \pi \cdot [\lambda_1, \ldots, \lambda_m]^T,$$

where π is the stationary vector of the modulating chain, i.e.:

$$\pi \mathbf{Q} = [0, \ldots, 0],$$
$$\pi \cdot \overrightarrow{1} = 1.$$

Hereafter $\overrightarrow{1}$ denotes the column vector of 1's.

The MMPP counting function is defined as:

$$P_{ij}(k, t) = \mathbb{P}(N(t) = k, J(t) = j | N(0) = 0, J(0) = i),$$

where $\mathbb{P}$ denotes probability, $N(t)$ denotes the number of events (jobs, customers, packets) in time interval $(0, t]$, while $J(t)$ denotes the state of the modulating chain at time t.

In this paper we pay a special attention to the autocorrelated structure of the MMPP. The $k-$lag autocorrelation of the MMPP equals:

$$Corr(k) = \frac{Cov(k)}{Var},$$

where

$$Cov(k) = p\,(\mathbf{\Lambda} - \mathbf{Q})^{-2}\mathbf{\Lambda}\left[[(\mathbf{\Lambda} - \mathbf{Q})^{-1}\mathbf{\Lambda}]^{k-1} - \overrightarrow{1}\,p\right](\mathbf{\Lambda} - \mathbf{Q})^{-2}\mathbf{\Lambda}\,\overrightarrow{1},$$

$$Var = 2\,p\,(\mathbf{\Lambda} - \mathbf{Q})^{-3}\mathbf{\Lambda} \cdot \overrightarrow{1} - \frac{1}{\lambda^2},$$

$$p = \frac{1}{\lambda}\pi\mathbf{\Lambda}.$$

4 Queueing Model

The system of interest is a single-server queue, in which the customers form the Markov-modulated Poisson process and they are served in the arrival order. A general type of service time distribution, given by a distribution function $F(t)$, is assumed. The queue size is limited by a finite buffer. Namely, the total number of jobs present in the system is always smaller or equal to b. A job that arrives when the buffer is full (i.e. there are b jobs present in the system) is dropped and never returns.

In addition to that, any arriving job can be dropped. This happens with probability $d(n)$, where n is the queue size at the time of arrival of this job (including the service position).

The function $d(n)$ is called the dropping function. It may take any values in $[0, 1]$ for $n = 0, \ldots, b - 1$. The finite-buffer assumption forces that $d(n) = 1$ for $n \geq b$.

The queue size at time t will be denoted by $X(t)$. We adopt the convention that $X(t)$ includes the service position, if occupied. If $X(0) > 0$, then it is assumed that the time origin corresponds to a service completion.

The load offered to the queue is then:

$$\rho = \lambda \int_0^\infty x dF(x).$$

5 Queue Size and Loss Ratio

Let us denote by $\Phi_{n,i}(t,l)$ the probability that the queue size at time t equals l, provided that the initial queue size was n and the initial state of the modulating chain was i, i.e.:

$$\Phi_{n,i}(t,l) = \mathbb{P}(X(t) = l | X(0) = n, J(0) = i),$$

where $0 \le n \le b,\ 1 \le i \le m,\ t > 0,\ 0 \le l \le b$.

In order to compute $\Phi_{n,i}(t,l)$, we have to use function $A_{n,k,i,j}(u)$, which is the counting function of the arrival process filtered by the dropping mechanism. Namely, $A_{n,k,i,j}(u)$ is defined as the probability that in a system without service (arrivals only), exactly k jobs are accepted to the queue in time interval $(0,u]$ and at the end of this interval the state of the modulating chain is j, provided that the queue size at $t = 0$ was n and the state of the modulating chain at $t = 0$ was i.

Assuming that the system is not empty at the time origin and using the total probability formula with respect to the first departure time, u, we obtain for $1 \le n \le b$, $1 \le i \le m$:

$$\Phi_{n,i}(t,l) = \sum_{j=1}^{m} \sum_{k=0}^{b-n} \int_0^t A_{n,k,i,j}(u) \Phi_{n+k-1,j}(t-u,l) dF(u) + \rho_{n,i}(t,l), \tag{1}$$

where

$$\rho_{n,i}(t,l) = \begin{cases} 0 & \text{if} \quad l < n, \\ (1 - F(t)) \sum_{j=1}^{m} A_{n,l-n,i,j}(t) & \text{if} \quad n \le l \le b. \end{cases}$$

The first summand of Eq. 1 corresponds to the situation, where the first service completion time, u, occurs before t, while the second summand to the situation where there is no service completion by the time t.

Assuming that the system is empty at the time origin and using the total probability formula with respect to the first event in the arrival process, which may be a change of the modulating state or a job arrival, we get for $1 \le i \le m$:

$$\begin{aligned} \Phi_{0,i}(t,l) \ = \ & \sum_{j=1}^{m} \int_0^t \Phi_{0,j}(t-u,l)(\lambda_i - \mathbf{Q}_{ii}) p_{ij} e^{-(\lambda_i - \mathbf{Q}_{ii})u} du \\ & + d(0) \sum_{j=1}^{m} \int_0^t \Phi_{0,j}(t-u,l) \mathbf{\Lambda}_{ij} e^{-(\lambda_i - \mathbf{Q}_{ii})u} du \\ & + (1 - d(0)) \sum_{j=1}^{m} \int_0^t \Phi_{1,j}(t-u,l) \mathbf{\Lambda}_{ij} e^{-(\lambda_i - \mathbf{Q}_{ii})u} du \\ & + \delta_{0l} e^{-(\lambda_i - \mathbf{Q}_{ii})t}. \end{aligned} \tag{2}$$

where

$$p_{ij} = \begin{cases} 0 & \text{if} \quad i = j, \\ \frac{\mathbf{Q}_{ij}}{\lambda_i - \mathbf{Q}_{ii}} & \text{if} \quad i \neq j, \end{cases}$$

$\mathbf{\Lambda}_{ij}$, $\mathbf{Q}_{ij}$ denote the entries in the i-th row and j-th column of $\mathbf{\Lambda}$, $\mathbf{Q}$, respectively, while $\delta_{ij} = 1$ if $i = j$ and 0 otherwise.

The first summand of Eq. 2 corresponds to the case, where the first event happens before t and it is a change of the modulating state. The second summand corresponds to the case, where the first event happens before t, it is a job arrival, but the job is dropped. The third summand corresponds to the case, where the first event happens before t, it is a job arrival and it is accepted. Finally, the fourth summand corresponds to the case, where there are no events in the MMPP by the time t.

Introducing the following notation:

$$\tilde{a}_{n,k,i,j}(s) = \int_0^\infty e^{-st} A_{n,k,i,j}(t) dF(t),$$

$$\overline{d}_{n,k,i,j}(s) = \int_0^\infty e^{-st} A_{n,k,i,j}(t)(1 - F(t))dt,$$

$$\overline{\phi}_{n,i}(s,l) = \int_0^\infty e^{-st} \Phi_{n,i}(t,l)dt,$$

and applying the Laplace transform to Eqs. 1 and 2 we get for $1 \leq n \leq b$, $1 \leq i \leq m$:

$$\overline{\phi}_{n,i}(s,l) = \sum_{j=1}^{m} \sum_{k=0}^{b-n} \tilde{a}_{n,k,i,j}(s) \overline{\phi}_{n+k-1,j}(s,l) + \int_0^\infty e^{-st} \rho_{n,i}(t,l)dt, \tag{3}$$

and for $1 \leq i \leq m$:

$$\overline{\phi}_{0,i}(s,l) = \sum_{j=1}^{m} \frac{(\lambda_i - \mathbf{Q}_{ii})p_{ij} + d(0)\mathbf{\Lambda}_{ij}}{s + \lambda_i - \mathbf{Q}_{ii}} \overline{\phi}_{0,j}(s,l) + \sum_{j=1}^{m} \frac{(1 - d(0))\mathbf{\Lambda}_{ij}}{s + \lambda_i - \mathbf{Q}_{ii}} \overline{\phi}_{1,j}(s,l) + \frac{\delta_{0,l}}{s + \lambda_i - \mathbf{Q}_{ii}}, \tag{4}$$

respectively. Using the following $m \times m$ matrices:

$$\mathbf{0} = [0]_{i,j}, \tag{5}$$

$$\tilde{\mathbf{A}}_{n,k}(s) = \left[\tilde{a}_{n,k,i,j}(s)\right]_{i,j}, \tag{6}$$

$$\overline{\mathbf{D}}_{n,k}(s) = \left[\overline{d}_{n,k,i,j}(s)\right]_{i,j}, \tag{7}$$

$$\mathbf{U}_n(s) = \left[\frac{(\lambda_i - \mathbf{Q}_{ii})p_{ij} + d(n)\mathbf{\Lambda}_{ij}}{s + \lambda_i - \mathbf{Q}_{ii}}\right]_{i,j}, \tag{8}$$

$$\mathbf{V}_n(s) = \left[\frac{(1 - d(n))\mathbf{\Lambda}_{ij}}{s + \lambda_i - \mathbf{Q}_{ii}}\right]_{i,j}, \tag{9}$$

and the following column vectors of size m:

$$\overline{\phi}_n(s,l) = [\overline{\phi}_{n,1}(s,l), \ldots, \overline{\phi}_{n,m}(s,l)]^T, \tag{10}$$

$$z(s) = \left[\frac{1}{s + \lambda_1 - \mathbf{Q}_{11}}, \ \cdots \ , \frac{1}{s + \lambda_m - \mathbf{Q}_{mm}}\right]^T, \tag{11}$$

$$r_n(s,l) = \begin{cases} \mathbf{0} \cdot \overrightarrow{1}, & \text{if} \quad l < n, \\ \overline{\mathbf{D}}_{n,l-n}(s) \cdot \overrightarrow{1}, & \text{if} \quad n \leq l \leq b. \end{cases} \tag{12}$$

we can rewrite Eqs. 3 and 4 as

$$\overline{\phi}_n(s,l) = \sum_{k=0}^{b-n} \tilde{\mathbf{A}}_{n,k}(s)\overline{\phi}_{n+k-1}(s,l) + r_n(s,l), \qquad 1 \le n \le b, \tag{13}$$

$$\overline{\phi}_0(s,l) = \mathbf{U}_0(s)\overline{\phi}_0(s,l) + \mathbf{V}_0(s)\overline{\phi}_1(s,l) + \delta_{0l}z(s), \tag{14}$$

respectively.

Then we may group all known coefficients of Eqs. 13 and 14 into one $(b+1)m \times (b+1)m$ matrix

$$\mathbf{G}(s) = \left[\mathbf{G}_{ij}(s)\right]_{i=0\ldots b, j=0\ldots,b}$$

in the form:

$$\mathbf{G}_{ij}(s) = \begin{cases} \tilde{\mathbf{A}}_{i,j-i+1}(s), & \text{if} \quad 1 \le i \le b-2, i < j \le b-1, \\ \tilde{\mathbf{A}}_{i,1}(s) - \mathbf{I}, & \text{if} \quad i = j, 1 \le i \le b-1, \\ \tilde{\mathbf{A}}_{i,0}(s), & \text{if} \quad i = j+1, \\ \mathbf{U}_0(s) - \mathbf{I}, & \text{if} \quad i = 0, j = 0, \\ \mathbf{V}_0(s), & \text{if} \quad i = 0, j = 1, \\ -\mathbf{I}, & \text{if} \quad i = b, j = b, \\ \mathbf{0}, & \text{otherwise}, \end{cases} \tag{15}$$

where $\mathbf{I}$ is the unit matrix of size $m \times m$. Now the system Eqs. 13 and 14 can be rewritten as

$$\mathbf{G}(s)\overline{\phi}(s,l) = R(s,l), \tag{16}$$

where $R(s,l)$, $\overline{\phi}(s,l)$ are the following column vectors of size $(b+1)m$:

$$\overline{\phi}(s,l) = \left[\overline{\phi}_0(s,l), \ldots, \overline{\phi}_b(s,l)\right]^T, \tag{17}$$

$$R(s,l) = [R_0(s,l), \ldots, R_b(s,l)]^T, \quad R_i(s,l) = \begin{cases} -\delta_{0l}z(s), & \text{if} \quad i = 0, \\ -r_i(s,l), & \text{if} \quad 1 \le i \le b. \end{cases} \tag{18}$$

In order to compute $\overline{\phi}(s,l)$ from Eq. 16, matrix $\mathbf{G}(s)$ must have an inverse. As it can be hard to prove that $\mathbf{G}(s)$ is non-singular in general, we assume from now on that this is the case. This is not a problem in practice - we never came accross a singular $\mathbf{G}(s)$, despite performing a large number of numerical computations for different system parameters.

Now Eq. 16 can be solved and we obtain the following result.

Theorem 1 *The Laplace transform of the queue size distribution at time t of a finite-buffer queue with the dropping function has the form:*

$$\overline{\phi}(s,l) = \boldsymbol{G}^{-1}(s)R(s,l), \quad 0 \le l \le b. \tag{19}$$

where $\boldsymbol{G}(s)$ and $R(s,l)$ are given in formulas Eqs. 15 *and* 18 *respectively.*

The practical applicability of Theorem 1 depends on our ability to compute function $A_{n,k,i,j}(u)$ – all other components are simple functions of the parameters of the queueing system.

Therefore the whole next section will be devoted to the computation of $A_{n,k,i,j}(u)$.

Now let us observe only, that having $A_{n,k,i,j}(u)$, we can compute several important characteristics of the queueing system using Theorem 1. Firstly, the stationary distribution of the queue size can be computed. Namely, denoting

$$P_l = \lim_{t\to\infty} \mathbb{P}(X(t) = l),$$

and using the well-known properties of the Laplace transform we have

$$P_l = \lim_{s\to 0+} s\left[\mathbf{G}^{-1}(s)R(s,l)\right]_1, \tag{20}$$

where $[\cdot]_1$ denotes the first element of a vector. In fact, any other element could be used in Eq. 20.

Secondly, we can obtain the transient distribution of the queue size, i.e. the distribution at an arbitrary chosen time t, which requires numerical inversion of the Laplace transform (19).

Thirdly, using Eq. 20 we can compute the loss ratio and throughput. The loss ratio, L, is defined as the long-run fraction of jobs which were dropped upon arrival. In a time interval of length T, where T is large, the number of finished jobs is approximately equal to:

$$\frac{(1-P_0)T}{\int_0^\infty x dF(x)}.$$

In the same interval, there are approximately λT new arrivals. Therefore, the fraction of accepted jobs (the throughput) must be:

$$\gamma = \frac{1-P_0}{\rho}, \tag{21}$$

while the loss ratio must be:

$$L = 1 - \frac{1-P_0}{\rho}. \tag{22}$$

6 Calculating A

In this section we deal with finding function $A_{n,k,i,j}(u)$, which is a crucial task for practical applicability of Theorem 1.

Firstly, let us consider the case, where in time interval $(0,u]$ at least one job is accepted to the queue, i.e. where $k \geq 1$ in $A_{n,k,i,j}(u)$. The first event in time interval $(0,u]$ may be a change of the modulating state, or an arrival of a job which is immediately dropped, or an arrival of a job which gets accepted. Therefore, for any $k \geq 1$, $0 \leq n \leq b$, $1 \leq i \leq m$, $1 \leq j \leq m$ we have:

$$\begin{aligned} A_{n,k,i,j}(u) = & \sum_{a=1}^{m} \int_0^u (\lambda_i - \mathbf{Q}_{ii}) p_{ia} e^{-(\lambda_i - \mathbf{Q}_{ii})v} A_{n,k,a,j}(u-v)dv \\ & + \sum_{a=1}^{m} \int_0^u d(n) \mathbf{\Lambda}_{ia} e^{-(\lambda_i - \mathbf{Q}_{ii})v} A_{n,k,a,j}(u-v)dv \\ & + \sum_{a=1}^{m} \int_0^u (1-d(n)) \mathbf{\Lambda}_{ia} e^{-(\lambda_i - \mathbf{Q}_{ii})v} A_{n+1,k-1,a,j}(u-v)dv. \end{aligned} \tag{23}$$

Secondly, assuming that there are no jobs accepted in $(0,u]$, the first event in $(0,u]$ may only be a change of the modulating state, or an arrival of a job which is immediately

dropped. Alternatively, there may be no events in $(0, u]$ at all. Summarizing, for any $0 \leq n \leq b$, $1 \leq i \leq m$, $1 \leq j \leq m$ we obtain:

$$\begin{aligned} A_{n,0,i,j}(u) = & \sum_{a=1}^{m} \int_0^u (\lambda_i - \mathbf{Q}_{ii}) p_{ia} e^{-(\lambda_i - \mathbf{Q}_{ii})v} A_{n,0,a,j}(u-v) dv \\ & + \sum_{a=1}^{m} \int_0^u d(n) \mathbf{\Lambda}_{ia} e^{-(\lambda_i - \mathbf{Q}_{ii})v} A_{n,0,a,j}(u-v) dv \\ & + \delta_{ij} e^{-(\lambda_i - \mathbf{Q}_{ii})u}. \end{aligned} \tag{24}$$

Applying the Laplace transform to Eqs. 23 and 24 we get for $k \geq 1$:

$$\overline{a}_{n,k,i,j}(s) = \sum_{a=1}^{m} \frac{(\lambda_i - \mathbf{Q}_{ii}) p_{ia} + d(n) \mathbf{\Lambda}_{ia}}{s + \lambda_i - \mathbf{Q}_{ii}} \overline{a}_{n,k,a,j}(s) + \sum_{a=1}^{m} \frac{(1 - d(n)) \mathbf{\Lambda}_{ia}}{s + \lambda_i - \mathbf{Q}_{ii}} \overline{a}_{n+1,k-1,a,j}(s). \tag{25}$$

and

$$\overline{a}_{n,0,i,j}(s) = \sum_{a=1}^{m} \frac{(\lambda_i - \mathbf{Q}_{ii}) p_{ia} + d(n) \mathbf{\Lambda}_{ia}}{s + \lambda_i - \mathbf{Q}_{ii}} \overline{a}_{n,0,a,j}(s) + \frac{\delta_{ij}}{s + \lambda_i - \mathbf{Q}_{ii}}, \tag{26}$$

where

$$\overline{a}_{n,k,i,j}(s) = \int_0^{\infty} e^{-st} A_{n,k,i,j}(u) du.$$

Using the following $m \times m$ matrices:

$$\overline{\mathbf{A}}_{n,k}(s) = \left[\overline{a}_{n,k,i,j}(s) \right]_{i,j},$$

$$\mathbf{Z}(s) = diag(z(s)),$$

for $k \geq 1$ (25) yields:

$$\overline{\mathbf{A}}_{n,k}(s) = \mathbf{U}_n(s) \overline{\mathbf{A}}_{n,k}(s) + \mathbf{V}_n(s) \overline{\mathbf{A}}_{n+1,k-1}(s), \tag{27}$$

while Eq. 26 yields:

$$\overline{\mathbf{A}}_{n,0}(s) = \mathbf{U}_n(s) \overline{\mathbf{A}}_{n,0}(s) + \mathbf{Z}(s). \tag{28}$$

Hereafter we assume that matrix $\mathbf{I} - \mathbf{U}_n(s)$ is non-singular. As in the case of $\mathbf{G}(s)$, we verified this assumption in a large number of numerical calculations for different system parameterizations. Denoting

$$\mathbf{C}_n(s) = (\mathbf{I} - \mathbf{U}_n(s))^{-1} \mathbf{V}_n(s), \tag{29}$$

and

$$\mathbf{E}_n(s) = (\mathbf{I} - \mathbf{U}_n(s))^{-1} \mathbf{Z}(s), \tag{30}$$

we have for $k \geq 1$:

$$\overline{\mathbf{A}}_{n,k}(s) = \mathbf{C}_n(s) \overline{\mathbf{A}}_{n+1,k-1}(s), \tag{31}$$

and

$$\overline{\mathbf{A}}_{n,0}(s) = \mathbf{E}_n(s). \tag{32}$$

Finally, exploiting Eqs. 31, 32 and the mathematical induction we arrive at the following theorem.

Theorem 2 *The Laplace transform of the counting function of the MMPP filtered by the dropping function is*

$$\overline{\boldsymbol{A}}_{n,k}(s) = \boldsymbol{C}_n(s)\boldsymbol{C}_{n+1}(s)\ldots\boldsymbol{C}_{n+k-1}(s)\boldsymbol{E}_{n+k}(s), \quad k \geq 1, \tag{33}$$

$$\overline{\boldsymbol{A}}_{n,0}(s) = \boldsymbol{E}_n(s), \tag{34}$$

where $\boldsymbol{C}_n(s)$ *and* $\boldsymbol{E}_n(s)$ *are given in* Eqs. 29 *and* 30*, respectively.*

It can be easily verified, that this is a generalization of the result for the ordinary Poisson process presented in Theorem 1 of Chydzinski and Chrost (2011). As for the numerical calculations of $A_{n,k,i,j}(u)$ values, we use the Spinelli inversion method (Spinelli 1966).

7 Examples

For numerical purposes the following MMPP parameterization will be used:

$$\mathbf{Q} = \begin{bmatrix} -172.53 & 38.80 & 30.85 & 0.88 & 102.00 \\ 16.76 & -883.26 & 97.52 & 398.9 & 370.08 \\ 281.48 & 445.97 & -1594.49 & 410.98 & 456.06 \\ 23.61 & 205.74 & 58.49 & -598.93 & 311.09 \\ 368.48 & 277.28 & 7.91 & 32.45 & -686.12 \end{bmatrix}, \tag{35}$$

$$[\lambda_1, \cdots, \lambda_5] = [59620.6, 113826.1, 7892.6, 123563.2, 55428.2].$$

This parameterization was obtained in Chydzinski (2006a) as a result of fitting the MMPP to the recorded network traffic. The autocorrelation function of this MMPP is shown in Fig. 3, which is to be compared with Fig. 2, depicting the autocorrelation of the original traffic. The average rate of this MMPP is $\lambda = 71729.36$.

In all examples the buffer size is $b = 200$. The service time is constant and chosen in such a way that the queueing system is mildly overloaded, i.e. $\rho = 1.1$.

Assume first that we want to obtain the system throughput of 80 %. Manipulating the shape of the dropping function and checking the resulting throughput by means of Theorem 1 and

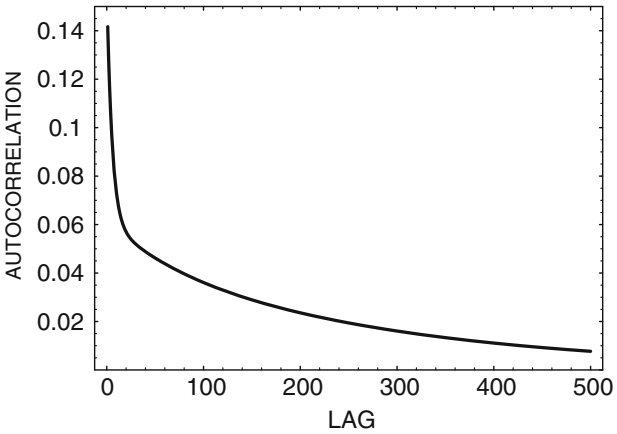

Fig. 3 Autocorrelation of the MMPP used in numerical examples

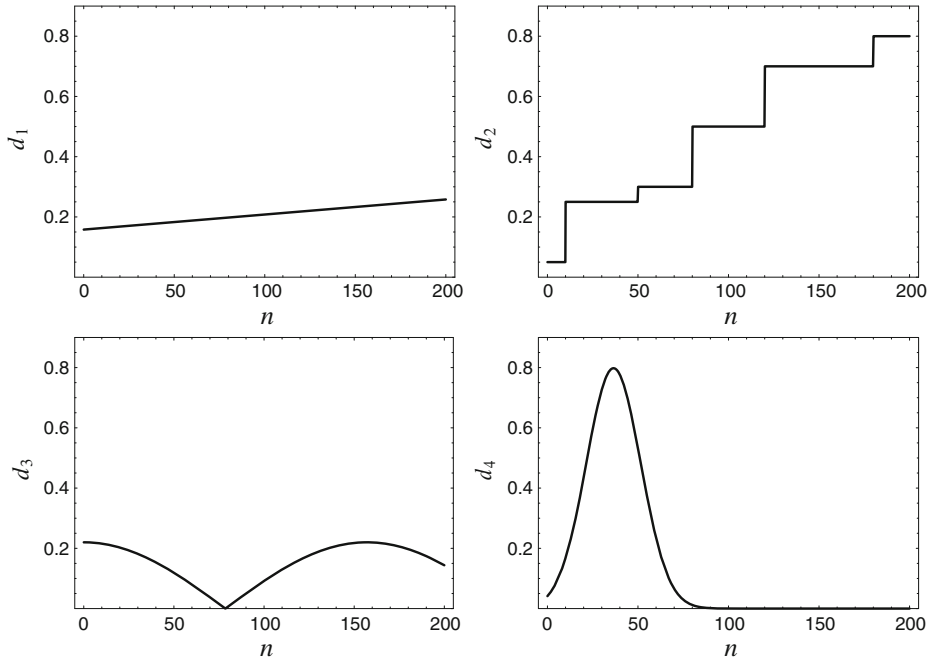

Fig. 4 Four different dropping functions providing the throughput of 80 %

formula (21) we can find many shapes suitable for our purposes. In particular, the following four dropping functions provide the throughput of 80 %:

$$d_1(n) = \begin{cases} 0.0005n + 0.158, & \text{for } 0 \le n < 200, \\ 1, & \text{for } n \ge 200, \end{cases}$$

$$d_2(n) = \begin{cases} 0.035, & \text{for } 0 \le n \le 10, \\ 0.25, & \text{for } 10 < n \le 50, \\ 0.30, & \text{for } 50 < n \le 80, \\ 0.50, & \text{for } 80 < n \le 120, \\ 0.70, & \text{for } 120 < n \le 180, \\ 0.80, & \text{for } 180 < n < 200, \\ 1, & \text{for } n \ge 200, \end{cases}$$

$$d_3(n) = \begin{cases} 0.22|\cos(n/50))|, & \text{for } 0 \le n < 200, \\ 1, & \text{for } n \ge 200, \end{cases}$$

Table 1 Performance characteristics of systems with dropping functions $d_1, \ldots, d_4$

dropping function	throughput, γ	average queue size	std. dev. queue size
d_1	80.0 %	48.5	61.6
d_2	80.0 %	4.53	3.45
d_3	80.0 %	64.7	67.8
d_4	80.0 %	7.39	7.94

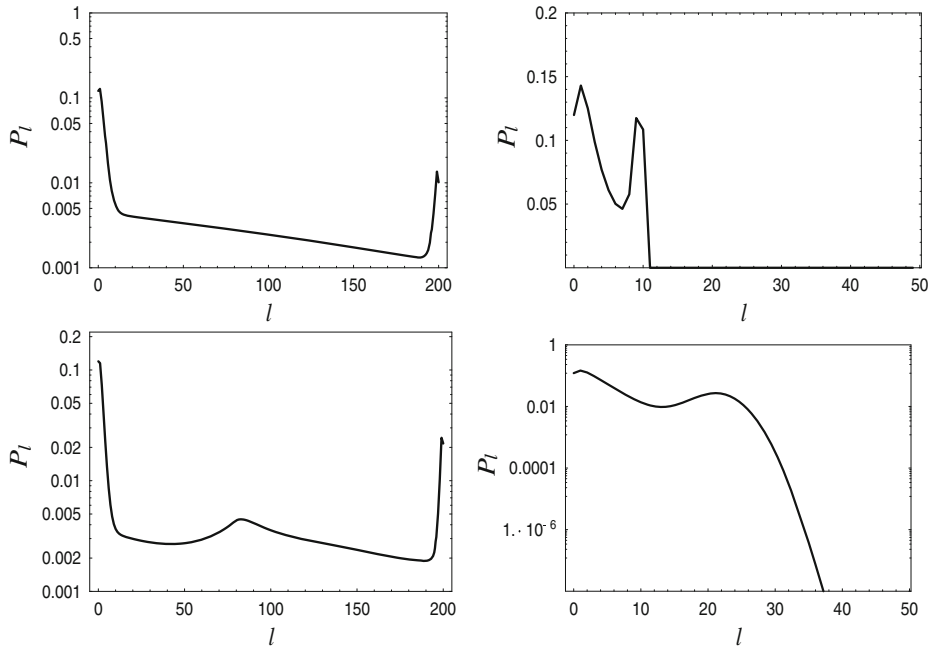

Fig. 5 Stationary queue size distributions for dropping functions: d_1 (*upper left*), d_2 (*upper right*), d_3 (*lower left*) and d_4 (*lower right*)

$$d_4(n) = \begin{cases} 0.797884561 e^{-(n-36.5)^2/450}, & \text{for } 0 \le n < 200, \\ 1, & \text{for } n \ge 200. \end{cases}$$

In Fig. 4 the shapes of dropping functions $d_1, \ldots, d_4$ are presented. The consequence of different shapes of these dropping function is that other than throughput performance characteristics differ significantly. The values of the average queue size and its standard

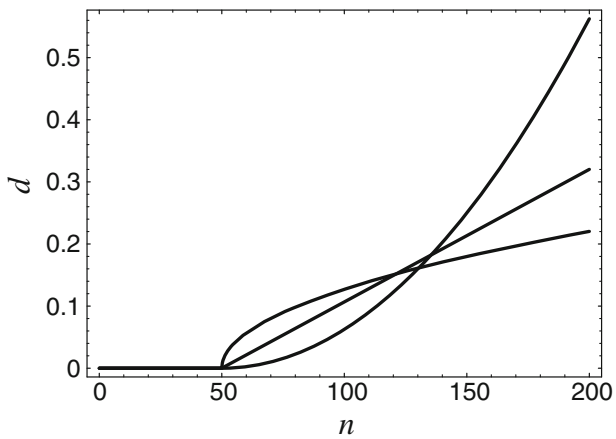

Fig. 6 Three different dropping functions providing the average queue size of 75

Table 2 Performance characteristics of systems with dropping functions d_5, d_6 and d_7

dropping function	throughput, γ	average queue size	std. dev. queue size
d_5	88.1 %	75.0	63.6
d_6	88.3 %	75.0	58.9
d_7	88.0 %	75.0	65.9

deviation are given in Table 1, while the stationary distributions of the queue size are shown in Fig. 5.

As we can see, the distributions are quite different and the average queue size can vary by an order of magnitude, depending on the shape of the dropping function, even though the throughput is kept at the level of 80 %. This indicate that we may search for dropping functions that provide both the required throughput and the required average queue size, e.g. $\gamma = 80$ % and the average queue $= 30$.

Finding dropping functions that provide only a predefined average queue size is very easy. For instance, the following three dropping functions assure the average queue size of 75:

$$d_5(n) = \begin{cases} 0, & \text{for } 0 \le n \le 50, \\ 0.002134228n - 0.106711409, & \text{for } 50 < n < 200, \\ 1, & \text{for } n \ge 200, \end{cases}$$

$$d_6(n) = \begin{cases} 0, & \text{for } 0 \le n \le 50, \\ 0.000025n^2, & \text{for } 50 < n < 200, \\ 1, & \text{for } n \ge 200, \end{cases}$$

$$d_7(n) = \begin{cases} 0, & \text{for } 0 \le n \le 50, \\ 0.017985611\sqrt{n - 50}, & \text{for } 50 < n < 200, \\ 1, & \text{for } n \ge 200. \end{cases}$$

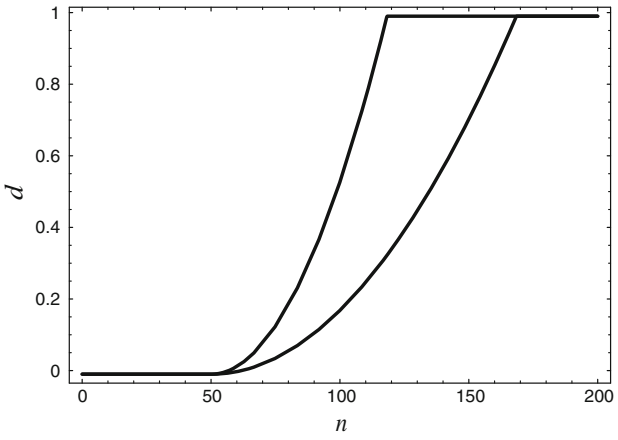

Fig. 7 Dropping functions providing the average queue size of 50 and 25, respectively

Table 3 Performance characteristics of systems with dropping functions d_8 and d_9

dropping function	throughput, γ	average queue size	std. dev. queue size
d_8	82.7 %	50.0	45.8
d_9	61.5 %	25.0	34.5

The shapes of functions d_5, d_6 and d_7 are presented in Fig. 6, while their main performance characteristics in Table 2. Now the values of the performance characteristics are close. This is connected with the fact that functions d_5–d_7 are all monotonic and positive for $n > 50$.

If we have dropping function d that provides the average queue size a, it is easy to obtain any average queue size smaller than a by scaling the function d, i.e. using:

$$d'(n) = \min\{c \cdot d(n), 1\},$$

where $c > 1$. For instance, using function d_6 we can obtain functions d_8 and d_9 which give the average queue of 50 and 25, respectively. Namely, we have:

$$d_8(n) = \min\{2.85 \cdot d_6(n), 1\},$$

$$d_9(n) = \min\{8.59 \cdot d_6(n), 1\},$$

see Fig. 7. The performance of these functions is summarized in Table 3.

By means of the dropping function we may also force particular values of performance characteristics for different load values. For instance, assume that we want the average queue size to be 60 for an overloaded system ($\rho = 1.1$) and 20 for an underloaded system ($\rho = 0.9$).

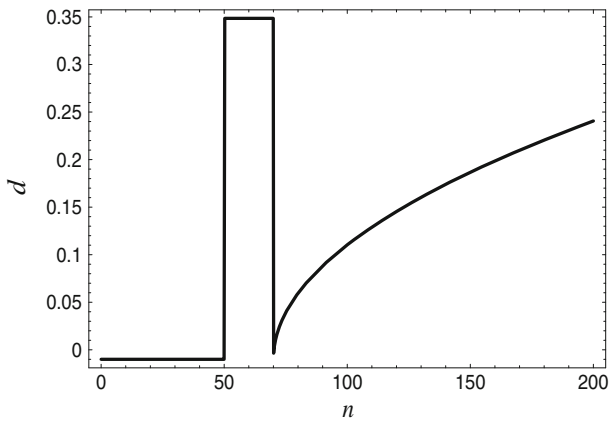

Fig. 8 Dropping function providing the average queue size of 60 for $\rho = 1.1$ and 20 for $\rho = 0.9$

Table 4 Performance characteristics of systems with dropping function d_{10} and two different loads

dropping function	throughput, γ	average queue size	std. dev. queue size
d_{10}, $\rho = 1.1$	87.3 %	60.0	61.9
d_{10}, $\rho = 0.9$	92.1%	20.0	32.3

Again, manipulating the shape of the dropping function we may achieve this goal. For instance, the following function meets the presented requirements:

$$d_{10}(n) = \begin{cases} 0, & \text{for } n \leq 50, \\ 0.3585, & \text{for } 50 < n \leq 70, \\ 0.02198\sqrt{n-70}, & \text{for } 70 < n < 200, \\ 1, & \text{for } n \geq 200. \end{cases}$$

Function d_{10} is depicted in Fig. 8, while its performance is summarized in Table 4.

It can be interesting to see, in one figure, the trade-off between the average queue size and throughput for different dropping functions. For this purpose, a scatter plot is presented in Fig. 9 for functions d_0-d_{10}, where d_0 is the zero dropping function, i.e.:

$$d_0(n) = \begin{cases} 0, & \text{if } n < 200, \\ 1, & \text{if } n \geq 200. \end{cases}$$

Of course, for d_0 we obtain the maximum value of the average queue size, 119.4, and the maximum value of the throughput, 89.5 %. As we can see, among the considered functions, d_2 provides the smallest average queue size, while maintaining a moderate throughput (only 9.5 % worse than the best possible). On the other hand, d_5 provides a very high throughput (only 1.4 % worse than the best possible) while maintaining a moderate average queue size.

It is also interesting to draw a scatter plot for several dropping functions belonging to one family. For instance, we studied a family of linear functions in the form:

$$d_k(n) = \begin{cases} an, & \text{if } n < 200 \text{ and } an < 1, \\ 1, & \text{otherwise}, \end{cases}$$

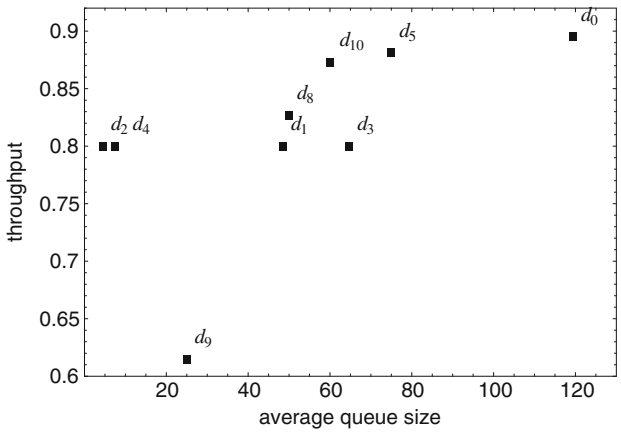

Fig. 9 The average queue size and throughput for dropping functions d_0-d_{10}

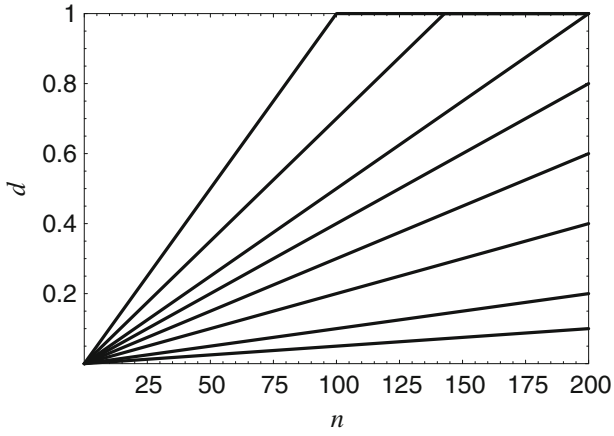

Fig. 10 Dropping functions d_{11}–d_{18}, counting from the bottom

where a is a parameter. The following values of a were used in computations: 0.0005, 0.001, 0.002, 0.003, 0.004, 0.005, 0.007, 0.01, for $k = 11, 12, \ldots, 18$, respectively. Therefore 8 new dropping functions, d_{11}–d_{18}, were obtained. They are depicted in Fig. 10. As we can see in Fig. 11, the steeper the function, the smaller the queue size, but the smaller the throughput at the same time. Therefore, we cannot say that one of the considered functions is "better" than other.

In this way, a very interesting question arises: what is the border of the set of all possible points in such scatter plots? In other words, what is the maximal possible throughput, given the average queue size of x? What is the minimal possible average queue size, given the throughput of y? So far, such questions seem to be hard to answer. Of course, using the results for d_0 we know that we can obtain any average queue size in interval [0, 119.4] and any throughput in interval [0, 89.5 %]. But it is hard to say for instance, if the average queue of 20.0 and the throughput of 89.0 % can be obtained at the same time.

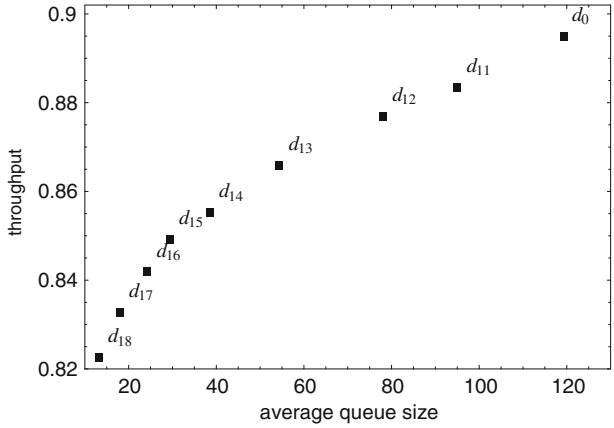

Fig. 11 The average queue size and throughput for dropping functions d_{11}-d_{18} and d_0

8 Conclusions

We presented an analysis of the queueing system with the dropping function and autocorrelated interarrival times. We argued that taking into account the autocorrelated structure of the arrival process is crucial for applicability of the model in some important areas of applications, e.g. in networking. The main results of the analysis are the transient and stationary distributions of the queue size and the stationary loss ratio and throughput in the model with the MMPP arrivals. We also presented several numerical results exploiting 18 different dropping functions, the buffer for 200 packets and a model of autocorrelated traffic recorded in an IP network. The main purpose of these examples was to demonstrate the practical usability of the model for solving systems with realistic parameterizations (properly parameterized traffic model, reasonable buffer size and load etc.).

References

Athuraliya S et al (2001) REM: Active queue management. IEEE Netw 15(3):48–53

Baiocchi A, Blefari-Melazzi N (1992) Steady-state analysis of the MMPP/G/1/K queue. IEEE Trans Commun 41(4):531–534

Bekker R (2009) Queues with Levy input and hysteretic control. Queueing Syst 63(1):281–299

Bonald T, May M, Bolot J-C (2000) Analytic evaluation of RED performance, Proceedings of INFOCOM, 1415–1424

Chydzinski A (2002) The M/G-G/1 oscillating queueing system. Queueing Systems 42(3):255–268

Chydzinski A (2006a) Transient analysis of the MMPP/g/1/k queue. Telecommun Syst 32(4):247–262

Chydzinski A (2006b) Queue size in a BMAP queue with finite buffer. Proceedings of next generation teletraffic and Wired/Wireless advanced networking '06. Lect Notes Comput Sci 4003:200–210

Chydzinski A (2006c) Duration of the buffer overflow period in a batch arrival queue. Perform Eval 63(4-5):493–508

Chydzinski A, Chrost L (2011) Analysis of AQM queues with queue-size based packet dropping. Int J Appl Math Comput Sci 21(3):567–577

Deng L, Mark J (1993) Parameter estimation for Markov modulated Poisson processes via the EM algorithm with time discretization. Telecommun Syst 1:321–338

Farzaneh N et al (2013) A novel congestion control protocol with AQM support for IP-based networks. Telecommun Syst 52(1):229–244

Feng W et al (2002) The BLUE active queue management algorithms. IEEE/ACM Trans Networking 10(4):513–528

Fischer W, Meier-Hellstern K (1992) The Markov-modulated poisson process (MMPP) cookbook. Perform Eval 18(2):149–171

Floyd S, Jacobson V (1993) Random early detection gateways for congestion avoidance. IEEE/ACM Trans Networking 1:397–413

Hao W, Wei Y (2005) Extended $GI^X/M/1/n$ Queueing Model for Evaluating the Performance of AQM Algorithms with Aggregate Traffic, Proceedings of ICCNMC, pp. 395–414, LNCS 3619

Hollot CV et al (2002) Analysis and design of controllers for AQM routers supporting TCP flows. IEEE Trans Autom Control 47(6):945–959

Kahe G, Jahangir A, Ebrahimi B (2013) AQM controller design for TCP networks based on a new control strategy, Telecommunication Systems. doi:10.1007/s11235-013-9859-y

Kempa W (2013a) On non-stationary queue-size distribution in a finite-buffer queue controlled by a dropping function. Proc. International Conference on Informatics, pp. 67–72, Spisska Nova Ves

Kempa W (2013b) A direct approach to transient queue-size distribution in a finite-buffer queue with AQM. Appl Math Inf Sci 7(3):909–915

Kunniyur SS, Srikant R (2004) An adaptive virtual queue (AVQ) algorithm for active queue management. IEEE/ACM Trans Networking 12(2):286–299

Na Z et al (2012) A novel adaptive traffic prediction AQM algorithm journal. Telecommun Syst 49(1):149–160

Pacheco A, Ribeiro H (2008) Consecutive customer losses in oscillating $GI^X/m//n$ systems with state dependent services rates. Ann Oper Res 162(1):143–158

Rosolen V, Bonaventure O, Leduc G (1999) A RED discard strategy for ATM networks and its performance evaluation with TCP/IP traffic. ACM SIGCOMM Comput Commun Rev 29(3):23–43

Ryden T (1996) An EM algorithm for parameter estimation in Markov modulated poisson processes. Comput Stat Data Anal 21:431–447

Salvador P, Valadas R, Pacheco A (2003) Multiscale fitting procedure using Markov modulated poisson processes. Telecommun Syst 23(1-2):123–148

Singh LN, Dattatreya GR (2004) A novel approach to parameter estimation in Markov-modulated poisson processes, Proc. IEEE Emerging Technologies Conference (ETC) Richardson, TX

Spinelli RA (1966) Numerical inversion of laplace transforms. SIAM J Num Anal 3:636–649

Suzer MH, Kang K-D, Basaran C (2012) Active queue management via event-driven feedback control. Comput Commun 35(4):517–529

Tikhonenko O, Kempa W (2013) Queue-size distribution in M/G/1-type system with bounded capacity and packet dropping. Commun Comput Inf Sci 356:177–186

Wu W, Ren Y, Shan X (2001) A self-configuring PI controller for active queue management. In: Asia-Pacific Conference on Communications (APCC), Japan

Wydrowski B, Zukerman M (2002) GREEN: An active queue management algorithm for a self managed Internet. In: Proceeding of IEEE international conference on communications ICC2002, vol 4, pp 2368–2372

Yoshihara T, Kasahara S, Takahashi Y (2001) Practical time-scale fitting of self-similar traffic with Markov-modulated poisson process. Telecommun Syst 17(1/2):185–211